

### REMARKS

The Office Action dated April 14, 2004 has been received and carefully noted. The following remarks are submitted as a full and complete response thereto.

Claims 17-29 are pending the present application and are respectfully submitted for consideration.

The Examiner has rejected claims 17 and 23 - 28 under 35 USC 102(b), alleging that these claims are anticipated by the Kleiman patent (United States Patent No. 5,515,538). The Applicant does not agree and respectfully traverses.

The Applicant notes that the Examiner has rejected claims in view of the Kleiman patent in previous Office Actions. The Applicant submits that all of the arguments presented against the Kleiman patent in the past are still valid. In particular, as the Applicant has noted before, the Kleiman patent does not discuss SMP (symmetric multi-processing), but discusses a method of handling interrupts. The Kleiman patent clearly does not enable the reader to implement an SMP system in the manner of the claimed invention, and therefore cannot possibly be held to anticipate the claims.

The Applicant also wishes to focus on three other major issues with respect to the anticipation rejection in view of the Kleiman patent. Specifically, the Applicant submits that:

1. the Examiner's rejection fails completely if the SunOS cannot be considered to be an IPC (inter-process control) message-passing operating system, having operating system calls which execute in both critical and non-critical areas. If the SunOS is

not such an operating system then the Kleiman patent cannot be considered to anticipate any of claims 17 or 23 – 28, because it is describing a completely different operating system which has no relevance to the claimed subject matter.

The Applicant will show hereinafter, that the SunOS is clearly not such an operating system;

2. the Examiner alleged that claims 17 and 23 – 28 are anticipated by the Kleiman patent despite the fact that it does not describe many of the steps in these claims (for example, the steps regarding the execution of OS calls in non-critical areas). The Applicant submits that the test for anticipation has not been satisfied by simply alleging that the missing steps are “inherent” as the Examiner has done; and
3. the Examiner made reference to column 11, line 39 through column 12, line 11 of the Kleiman patent, as well as column 12, lines 53 – 55, as the basis for the rejection of claim 17. The Applicant notes that these sections of the Kleiman patent are not addressing the claimed SMP (symmetric multi processing) subject matter at all – they are very clearly referring to a manner of handling interrupts. Clearly, these portions of the Kleiman patent cannot be said to anticipate the claims if they are not even addressing the same subject matter.

To elaborate on each of these points:

### **On Message-Passing Operating Systems**

In response to the previous Office Action, the Applicant argued that the SunOS is not a message-passing operating system. The Examiner disputed these arguments in the current Final Action at items 14 – 16, offering the Powell USENIX publication as evidence that the SunOS is such an operating system.

The Applicant submits that the SunOS is a monolithic operating system, and not a message-passing operating system. While the USENIX article referred to by the Examiner does describe the use of kernel threads, this is clearly not new in the art.

The SunOS is a monolithic kernel, so there is no message passing for threads inside the operating system. Microkernel architectures use message passing for operating system threads, while monolithic operating systems do not. Monolithic operating systems have addressability between all components of the operating system, so message passing is not required. The Applicant submits that there is ample evidence of the SunOS's characterization in the industry:

1. under Tab A of the attached, the Applicant has included six pages from the following website: <http://cbbrowne.com/info/microkernel.html>. Christopher Browne is one of the co-authors of the book "Professional Linux Programming." Mr. Browne explains that messages and IPC mechanisms are used by microkernel operating systems (such as that of the claimed invention) and are not used by monolithic operating systems (such as that of the SunOS):

- a. In the first paragraph of the first page, he identifies the “**messages**” and “**interprocess communication (IPC)**” mechanisms used by microkernel operating systems (emphasis added);
- b. In the second paragraph of the first page, he notes that these mechanisms result in modularity and its subsequent advantages which are “not typically found in monolithic or conventional operating systems;”
- c. In the third paragraph of the first page, he notes that “microkernels move many of the OS services into ‘user space’ that on other operating systems are kept in the kernel;”
- d. Through the course of this document, a number of microkernel operating systems are identified, specifically: Mach, L4 and QNX. There is no mention of the SunOS as a microkernel architecture in this document;
- e. At the bottom of the second page, in contrasting microkernel architectures to monolithic architectures, he notes that “communications between components of the extended ‘OS’ requires that **formalized message-passing mechanisms** be used” (emphasis added).
- f. In section 3.1 on the fourth page, he explains that the “monolithic architecture implements all operating system abstractions in kernel space,” while the “microkernel architecture abstracts lower-level OS facilities, implementing them in kernel space, and moves higher-level facilities to processes in user space.”

Microkernel architectures require message passing at the OS level because the higher-level facilities are not implemented in the kernel. Monolithic architectures do not require message passing at the OS level because their OS calls are executed in the kernel. The fact that the SunOS monolithic kernel is using OS threads to gain some of the advantages of a microkernel is not going to change this.

2. under Tab B, a printout from the OSNews.com website found on the Internet at:

[http://www.osnews.com/story.php?news\\_id=7162](http://www.osnews.com/story.php?news_id=7162) is provided. This web page explains that:

- a. JunOS, IOS-XR and Mach are microkernel operating systems. There is no mention of the SunOS in this document;
- b. In the second entry on the third page, Rayiner Hashem explains his understanding that a pure microkernel requires all operating system processes to be “separate from the primary kernel and **use message passing...**” (emphasis added).

3. Andy Tanenbaum is a highly respected computer scientist and operating system expert, as well as the designer and implementer of the MINIX microkernel operating system. On his webpage at: <http://www.cs.vu.nl/~ast/brown/>, he presents an interview he had with Ken Brown, on the development and history of Linux. HTML and text copies of this interview are attached under Tab C.

Towards the end of this interview, Mr. Tanenbaum explains that QNX and MINIX are examples of microkernel operating systems, while Linux and other similar operating systems are monolithic;

4. a number of searches were done on the Sun website at: [http://onesearch.sun.com/search/developers/index.jsp?qt=solaris&col=javadoc&col=javatecharticles&col=java\\_tutorials&col=devarchive&col=javasc&col=devall](http://onesearch.sun.com/search/developers/index.jsp?qt=solaris&col=javadoc&col=javatecharticles&col=java_tutorials&col=devarchive&col=javasc&col=devall), and no evidence could be found of the SunOS or Solaris operating systems being described as anything but monolithic operating systems;
5. Pieter Dumon's article titled "OS Kernels – a little overview and comparison" is attached under Tab D, and is available online at <http://tunes.org/~unios/oskernels.html>. On page 3 of the attachment, he indicates that Solaris uses a monolithic architecture, while Mach, L4 and QNX are microkernel architectures.

At the bottom of page 4, he also explains that Mach uses message-passing;

6. Geoffrey Voelker's University of California in San Diego, slides titled "CSE 120 – Principles of Operating Systems – Fall 2001" is attached under Tab E, and is available online at <http://www.cs.ucsd.edu/classes/fa01/cse120/lectures/struct-bw.pdf>.

Slide 16 clearly identifies Solaris as having a monolithic kernel architecture, while slide 21 identifies Mach and Chorus as examples of microkernel architectures. It is the nature of the microkernel architecture (which is presented in slides 21 and 22) which

requires IPC message passing of OS calls. This is in direct contrast to the monolithic architecture shown in slide 16;

7. Tab F presents notes prepared by Gregory D. Benson, Assistant Professor in the Department of Computer Science at University of San Francisco. These notes are available online at: <http://www.cs.usfca.edu/benson/cs326/09-osdesign-notes.pdf>.

On page 3 of these notes, Solaris is clearly identified as having a monolithic kernel architecture (see item 09-8), while MINIX and Mach are identified as having microkernel architectures (see item 09-10). As noted under item 09-12, “all processes communicate using **message passing**” (emphasis added) for the MINIX OS, and all of the operating system functionality operates as a separate process except for the lowest level functions (process management and message passing itself). The balance of the notes elaborate on this;

8. Tab G presents the article titled “Toward Real Microkernels” published in the "Communications of the ACM" in September 1996, clearly identifies Solaris as a “nonmicrokernel system” (see page 75, left column), in contrast to Mach and Chorus.

Thus, SunOS and Solaris (basically, different versions of the same operating system) are monolithic kernel-based operating systems, which may use some operating system threads. However, they do not use message-passing internal to the operating system. As well, because they are monolithic, they do not have an inherent separation between critical and non-critical operating areas of the operating system.

### **On the Test for Anticipation:**

The Examiner alleged that a number of the claims are anticipated by the Kleiman patent, despite the fact that the Kleiman patent does not disclose or suggest the steps regarding execution of OS calls in non-critical areas. The Applicant submits that this does not satisfy the requirements for the anticipation test.

The Applicant notes that the *Manual of Patent Examination Procedure* reads as follows:

*"A claim is anticipated only if each and every element as set forth in the claim is found, either **expressly** or **inherently** described, in a single prior art reference."*

*Verdegaal Bros. v. Union Oil Co. of California, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). "The identical invention must be shown in as **complete detail** as is contained in the ... claim." Richardson v. Suzuki Motor Co., 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). (emphasis added)*

Clearly, the elements of the claims are not "expressly" described in the Kleiman patent, as the Examiner has not even made such an assertion.

The Examiner has alleged that a number of the claim elements are "inherent" in the Kleiman patent. The Examiner has alleged that the Kleiman patent describes the steps regarding the call to "critical" areas of the operating system, but that it is not necessary that it disclose or suggest the calls to non-critical areas. The Applicant does not agree.



The Applicant submits that it is not “inherent” in the operating system described in the Kleiman patent or in a typical operating system, that operating system calls would require access to both critical and non-critical areas. In order to have a system in which operating system calls may execute in both critical and non-critical areas, one requires at least the following:

- the functionality to clearly define when calls are going to which areas, and
- the functionality to ensure that there are no consistency problems.

This is very difficult to do, and is the primary reason that most operating systems are designed to execute only in critical areas.

Some operating systems do access both critical and non-critical areas, but not in the context of an operating system call. This can be shown in the context of the “garbage collection” example raised by the Examiner.

To begin with, garbage collection is a sub-service. When garbage collection is performed, if you have a critical operating system call, then, the non-critical garbage collection sub-service will continue to run, causing the operating system call to be blocked, waiting for the garbage collection to be completed (it has to be blocked for consistency). Thus, the operating system call itself, always executes in a critical area – it does not execute in a non-critical area at all.

These arguments apply to all three of the sub-services described by the Examiner. Monolithic operating systems consider all OS calls to be critical, and lock out other calls until they are handled. Most operating systems – and monolithic operating systems in

particular – do not have critical and non-critical areas. There is nothing “inherent” about such functionality at all.

Moreover, there is certainly nothing in the Kleiman patent which would teach the reader how to identify OS calls as being to critical or non-critical areas, and how to handle them. As noted above, the test for anticipation requires that “complete detail” be provided by the prior art. This certainly has not been done as there is no disclosure or suggestion at all in the Kleiman patent, on how to do this.

**On the Portions of the Kleiman Reference Which Were Cited:**

In rejecting claim 17, the Examiner made reference to column 11, line 39 through column 12, line 11 of the Kleiman patent, as well as column 12, lines 53 – 55. The Applicant notes that these sections of the Kleiman patent are not addressing the claimed SMP (symmetric multi processing) subject matter at all – they are very clearly referring to a manner of handling interrupts.

As the Applicant has noted in the past, the Kleiman patent discusses the problem of an interrupt causing a current thread context to have to be saved so the interrupt can be serviced. The Kleiman patent does not discuss how operating system calls can be executed more efficiently in an SMP environment, which is the subject matter of the present patent application. Thus, one skilled in the art would not look to the Kleiman patent for assistance in addressing the problems of the invention because Kleiman is dealing with completely different subject matter - improving the efficiency of their handling of interrupts, as opposed to targeting the OS calls.

A review the lines of text cited by the Examiner also bears this out. There is no mention of SMP at all, but a continuous discussion of how to handle interrupts.

The Applicant submits that the lines of text cited by the Examiner certainly do provide the “complete detail” required by the test for anticipation.

### **Conclusion Regarding Rejection of Claims 17 and 23 - 28**

Claims 23 - 26 include the same limitations as claim 17, but claim the invention in different forms (specifically, these are system, apparatus, memory medium and signal claims respectively).

Claim 27 depends from claim 17 and includes all of its limitations, but also adds the limitation that the critical area of the message-passing operating system is limited to the message passing operation. As noted above, the Kleiman patent does not describe an operating system which uses internal message-passing.

Claim 28 depends from claim 27 and includes all of the limitations of claims 17 and 27. It also adds the limitation of a second message-passing operation to complement that of claim 27. The Applicant submits that a similar argument to that expressed with respect to claim 27 also applies here.

The Applicant respectfully requests that the rejection of claims 17 and 23 - 28 under 35 USC 102 (b) be withdrawn.

### **REGARDING REJECTION OF CLAIMS 18 – 20 AND 29**

The Examiner then rejected claims 18 - 20 and 29 under 35 USC 103(a), alleging that these claims are obvious in view of the Kleiman patent in combination with Dangelo

(United States Patent No. 5,946,487). Again, the Applicant does not agree and respectfully traverses.

As a matter of background, the Applicant notes that the title of the Dangelo patent is "Object- Oriented Multi-Media Architecture." The abstract of the Dangelo patent describes the subject matter of the patent as: "An object-oriented, multi-media architecture provides for real-time processing of an incoming stream of pseudo-language byte codes compiled from an object-oriented source program."

The Examiner alleged that Dangelo shows that micro kernel operating systems could be used in the application of the invention, but this argument fails for a number of reasons. Most important is that Dangelo's definition of a "micro kernel operating system" is far different than the definition generally held in the art, and his definition certainly does not fall within that outlined in the specification (see page 6, line 34 through page 7, line 2 which reads that "A micro kernel operating system is one in <which> the operating system itself provides minimal services which delegate the usual operating system functions to external processes.") At lines 35 - 43 of column 9, Dangelo clearly defines his "micro kernel" operating system as what is known as a "monolithic" operating system, as it does far more internal processing: "The micro-kernel also attends to handling the network file system (NFS), networking operations, peripheral device drivers, virtual memory management, user interface, and other tasks for which the operating system conventionally is responsible." This is simply not a micro-kernel

operating system as known in the art, and as defined in the specification of the present application.

The Dangelo patent describes mutual exclusion locks or mutexes, but mutexes have been known in the art for many years - mutexes are standard POSIX thread-level synchronization primitives, like the semaphores addressed in response to the previous Office Action.

Mutexes are used to ensure exclusive access to data shared between threads - not to protect operating system calls. The data being accessed is stored in memory, so the mutexes protect that memory; in contrast, the locks used in the claimed invention protect operating system calls executing on a microprocessor - a completely different concept.

Redesigning an operating system to use mutex-like functionality for operating system calls would still not lead one to the claimed invention. The invention lies in the strategic use of locks for only part of the operating system call – the part within the critical areas of the operating system call. This is what provides the ability to overlap non-critical areas of operating system calls as shown in Figure 6 of the subject patent application.

While Dangelo may describe mutexes in the context of a microkernel operating system, it does not describe any of the functionality necessary to implement the invention of claim 18, either independently or in view of the Kleiman reference.

Thus, claim 18 cannot be considered obvious in view of the cited combination of references. The Kleiman patent does not describe all of the limitations of the parent

claim, claim 17, and the Dangelo patent does not describe the missing limitations.

Further, neither of these references describe the limitations of claim 18.

The Applicant submits that claims 19, 20 and 29 distinguish over the cited references in the same manner as described above. The Applicant therefore asks that the rejection of claims 18 – 20 and 29 be withdrawn.

#### **REGARDING REJECTION OF CLAIMS 21 AND 22**

The Examiner also rejected claims 21 and 22 in view of the Kleiman patent in combination with Dangelo and Jones et al. (United States Patent No. 5,812,844). The Applicant does not agree and respectfully traverses.

The Applicant notes that claims 21 and 22 depend from claims 17, 18, 19 and 20. Therefore, claims 21 and 22 include all of the limitations of claims 17 - 20. The Jones patent does nothing to address the limitations of claims 17 - 20, with respect to which the Kleiman and Dangelo patents are lacking. Because the Kleiman, Dangelo and Jones patents do not describe limitations of the parent claims, the Applicant submits that the dependent claims cannot be considered obvious in view of the same combination of references.

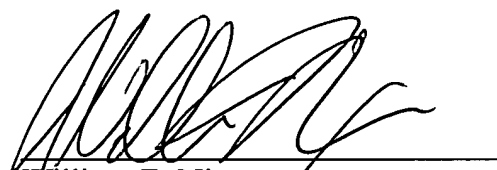
In view of the above amendments and remarks and having dealt with all the objections and rejections raised by the Examiner, reconsideration and allowance of the subject patent application is courteously requested.

Therefore, it is submitted that each of claims 17-29 recite subject matter that is neither disclosed nor suggested in the cited prior art. It is respectfully requested that all of claims 17-29 be allowed, and that this application passed to issue.

If for any reason the Examiner determines that the application is not now in condition for allowance, it is respectfully requested that the Examiner contact, by telephone, the applicant's undersigned attorney at the indicated telephone number to arrange for an interview to expedite the disposition of this application.

In the event this paper is not being timely filed, the applicant respectfully petitions for an appropriate extension of time. Any fees for such an extension together with any additional fees may be charged to Counsel's Deposit Account 50-2222.

Respectfully submitted,

  
\_\_\_\_\_  
William F. Nixon  
Registration No. 44,262

**Customer No. 32294**  
**SQUIRE, SANDERS & DEMPSEY LLP**  
14<sup>TH</sup> Floor  
8000 Towers Crescent Drive  
Tysons Corner, Virginia 22182-2700  
Telephone: 703-720-7800  
Fax: 703-720-7802

WFN:cct